Research trends in Multicore Programming

Contents of Lecture 12

- Power Efficient HTM
- Thread-level speculation
- C11Tester

Energy Efficient HTM

- By Do and Dubois from USC in Los Angeles 2016
- They found that over 42 % of aborted transactions have multiple aborts
- By delaying the restarting of aborted transaction, energy can be saved
- To each core a table is added which helps predict when it is better to delay a restart
- They found that up to 37 % energy could be saved this way
- `https://dl.acm.org/doi/10.1145/2875425`

HTM for speculative parallelization of for-loops

- By Salamanca, Amaral och Araujo, 2016 IEEE Trans par. distr. comp.
- Thread-level speculation (TLS) implemented with HTM
- Some loops cannot be analyzed by parallelizing compilers
- In software thread-level speculation normal hardware is used and data-races are detected and then the parallel execution of a loop is aborted
- This is normally difficult to make efficient

Evaluation on SPEC CPU2006

- SPEC CPU2006 contains single-threaded C/C++ and Fortran benchmarks
- It is an industry standard to evaluate compilers and CPUs
- First version from 1989 and current from 2017
- Some benchmarks were modified manually to run loops speculatively in parallel
- Four Intel and POWER8 CPUs were used and speedup of up to 3.8 was noted
- `https://ieeexplore.ieee.org/document/8038067`

A bug detector for C/C++ atomics

- There are no data-races between accesses to atomic variables
- How can bugs in lock-free data structures easily be found?
- Writing a few test cases is clearly insufficient
- Use stress tests which randomizes input and runs for hours.
- Is that sufficient?

A more systematic approach

- Evaluate all executions!
- Obviously not practical for millions of randomized inputs.
- Only a small number of test cases is realistic
- But how can all executions be evaluated???

C11Tester from UC Irvine

- The C11Tester is a library linked into the executable program
- Clang/LLVM instruments the application program with calls to C11Tester
- C11Tester has own functions for threads, mutex and atomics
- At runtime, C11Tester creates a graph showing the modification order of atomic variables
- C11Tester schedules the threads

Using the modification order graph

- The graph can be used to see which executions have not yet been tested
- The graph could quickly become extremely huge
- The key idea is to remove parts of the graph using semantics of atomic operations
- Using the graph, C11Tester can guarantee that all executions are actually tested
- The idea then is to find executions which triggers bugs
- Compare this to running the program millions of times until it crashes
- C11Tester can give much more useful information

`http://plrg.ics.uci.edu/c11tester`